تلميحات كتابة برمجيات نظيفة

في هذه المقالة قمت بذكر بعض التلميحات لكتابة تعليمات برمجية نظيفة

بحيث تسهل على كل مبرمج يتبعها بالرجوع إلى برامجه وفهم تعليماته البرمجية

أو في حال قرأ أي مبرمج آخر تلك التعليمات يفهمها بكل سهولة

الدوال:

1 –أسماء الدوال يجب أن تتكون من فعل واسم يصفان هدف هذه الدالة، ويجب كتابة الحرف الأول من كل كلمة بحرف كبير **Capital letter**. كمثال: `PrintStudentList()` بدلاً من `Print()`

2 –الدالة يجب أن يكون لها هدف وحيد، مثلاُ حساب معدلات الطلاب وطباعة أسماء الطلاب والحصول على الإدخالات بدالة واحدة أم ثلاث دوال منفصلة.

3 –الدوال التي تعيد قيمة يجب أن تعيد تلك القيمة من خلال عبارة **return** وليس من خلال قائمة المعاملات (**parameters**) الممررة.

4 –الدالة التي لا يمكن أن تتسع في صفحة مطبوعة واحدة على الأرجح أنها معقدة وطويلة ويجب أن يتم تقسيمها إلى أقسام أصغر.

5 –كل دالة يجب أن تحوي قسم في رأسها يحوي شرح عن عملها و عن المتغيرات التي يجب تمريرها و كل القيم المحتمل إعادتها، وبذلك فإن أي شخص يقرأ تلك الدالة ليس مضطراً إلى قراءة الدالة بالكامل وتتبع تنفيذها ليفهم عملها، مثال:

```
/* هذه الدالة تأخذ 3 معاملات للدخل: مصفوفة تحوي الأرقام التسلسلية للطلاب
 * ومصفوفة تحوي علامات الطلاب والمعامل الثالث هو الرقم المراد البحث عنه
 * وبعدها تقوم بالبحث في المصفوفة الأولى حسب الرقم المدخل وإذا لم يتم إيجاد
 * أي نتيجة ستعيد الدالة القيمة -1 وإلا فإنها ستعيد علامة الطالب الموجود
 */
```

التعليمات البرمجية:

1 –استخدم مقداراً كافياً من المسافات البيضاء (**Enter, Space, Tab**) لجعل التعليمات البرمجية واضحة.

2 –ضع التعليمات التي تنتهي بفاصلة منقوطة و أقواس {} بأسطر منفصلة (باستثناء الحلقات التكرارية)، مثال:

```
for (i=o;i<5;i++)
{ cout<<i*j; j=j-3; }
```

بينما الأفضل

```
for (i = 0; i < 5; i++)
{
cout << i * j;
j = j - 3;
}
```

فحص الأخطاء:

1 –أعط المستخدم معلومات مفيدة في حال حصل خطأ معين بالبرنامج، ولا تقم بإخافته بوضع عبارات مفاجئة ومربكة مثل: "قيمة غير صحيحة" بينما يفضل: "الرجاء إدخال أرقام بين 1 و 10".

2 –تأكد من أن البرنامج خالي من الأخطاء الداخلية الممكنة مثل القسمة على الرقم 0 ، نسخ النصوص أو تقسيم الأحرف في النصوص التي تحوي القيمة **NULL** ، الإشارة إلى عناصر خارج حدود المصفوفة ...الخ.

3 –حصن برنامجك من الاستخدام السيئ، وذلك بجعله آلياً قدر الإمكان، وكل الإدخالات التي تصل من مصادر خارجية يجب أن يتم فحصها والتأكد من صلاحيتها، مثال التأكد من أن الرقم المدخل بين قيمتين محددتين، أو أن الملف المدخل موجود قبل البدء بقراءته.

المتحولات / الثوابت:

1 –اختر أسماء المتحولات والثوابت بحذر، مثال:

`score = total_points - points_missed;` بدلاً من `c = a - b;`

2 –عرف كل المتغيرات والثوابت في قمة الدالة.

3 –المتغيرات بشكل عام لا يجب أن يتم شحنها بقيم يتم حسابها بنفس السطر الذي عرفت فيه، وبالرغم من ذلك فإن شحنها بقيم ثابتة مقبول، مثال: `int c = 0;` بدلاً من `int c = sqrt(b);`

4 –ضع تعليقاً لكل متغير قمت بتعريفه لوصف عمله.

5 –بشكل عام، يجب أن تكون أسماء كل الثوابت بأحرف كبيرة وأسماء المتغيرات بأحرف صغيرة، مثال:

```
int num_of_students = 20; // or numOfStudents
```

و

```
const int NAME_LEN = 4;
```

6 – استخدم الثوابت لكل الأرقام السحرية مثل قيمة **PI** (π) أو من أجل كل قيمة يمكن أن يتم استخدامها أكثر من مرة.

التوثيق:

1 –اكتب تعليقات تصف لماذا تمت العملية أكثر من كيف تمت، مثال:

```
تقسيم مجموع العلامات على عدد الطلاب //
```

بينا الأفضل:

```
حساب قيمة متوسط علامات الطلاب //
```

2 –دائما ضع اسمك، التاريخ، ووصف البرنامج في أعلى المستند البرمجي (البرنامج).

3 –اجعل تعليقاتك تقول شيئاً عن العمليات البرمجية حيث أنها لا تستطيع أن تشرح نفسها، مثال:

```
level++; // Add 1 to level
```

بينما الأفضل:

```
level++; // Advance to next level before processing remaining lives
```

4 –نموذجياً، التعليق يشرح مجموعة منطقية من الأسطر البرمجية أفضل من التعليق الذي يشرح سطر واحد.

5 –تجنب كل أو معظم الاختصارات عند كتابة التعليقات حتى تكون واضحة ومفهومة.

6 –قم بفصل التعليمات البرمجية عن التعليقات بشكل واضح.

7 –استخدم نمط تعليقات بحيث لا يكون ممل بشكل مفرط أو مضيع للوقت.

8 – مهد للبرنامج بتعليقات قبل البدء بكتابته أفضل من أن تقوم بكل عمليات التوثيق في النهاية، وبالتالي ستحمي وقتك من وجوب إيجاد الأماكن الدقيقة أو نسيان بعض التفاصيل، أو الافتراضات، ودقة التصميم.

النهاية

أتمنى أن تكون المقالة مفيدة ونالت استحسانكم

الكاتب: **Fuad Prince**

Fuad-prince@hotmail.com

www.svsonline.org

www.AhlamSoft.wordpress.com